

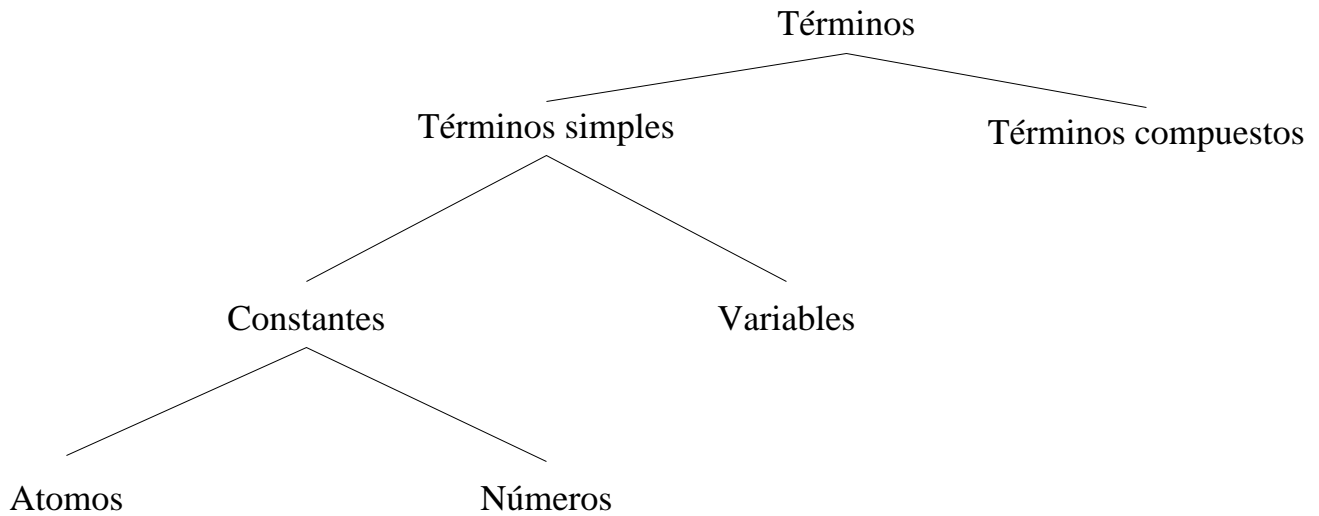
Tema 5: Otros predicados predefinidos

José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo

Dpto. de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE SEVILLA

Clasificación de términos



● Números

● Número: number

?- number(5.468).

Yes

?- number(-4).

Yes

?- number(a).

No

● Enteros: integer

?- integer(4).

Yes

?- integer(2.345).

No

Clasificación de términos

- **Punto flotante:** float

```
?- float(3.702).
```

```
Yes
```

```
?- float(-7).
```

```
No
```

- **Redondeo:** round

```
?- help(round).
```

```
round(+Expr)
```

```
    Evaluates Expr and rounds the result  
    to the nearest integer.
```

```
Yes
```

```
?- X is round(5.3).
```

```
X = 5
```

```
Yes
```

```
?- X is round(5.8).
```

```
X = 6
```

```
Yes
```

```
?- X is round(127/10).
```

```
X = 13
```

```
Yes
```

```
?- X is round((127/10)*1.34).
```

```
X = 17
```

```
Yes
```

```
?- X is round(5.5).
```

```
X = 6
```

```
Yes
```

```
?- X is round(5.4999).
```

```
X = 5
```

```
Yes
```

Clasificación de términos

- **Suma segura**

- Definir `suma_segura(X,Y,Z)` que se verifique si `X` e `Y` son enteros y `Z` es la suma de `X` e `Y`

- **Ejemplo:**

```
?- suma_segura(2,3,X).
```

```
X = 5
```

```
Yes
```

- **Programa `suma_segura.pl`**

```
suma_segura(X,Y,Z) :-  
    integer(X),  
    integer(Y),  
    Z is X+Y.
```

- **Sesión**

```
?- suma_segura(7,a,X).
```

```
No
```

```
?- X is 7 + a.
```

```
[WARNING: Arithmetic: 'a' is not a function]
```

```
^ Exception: ( 7) _G132 is 7+a ?
```

Clasificación de términos

- **Átomos y variables:** atom, atomic, var y nonvar

```
?- help(atom).
```

```
atom(+Term)
```

```
Succeeds if Term is bound to an atom.
```

```
Yes
```

```
?- help(atomic).
```

```
atomic(+Term)
```

```
Succeeds if Term is bound to an atom,  
string, integer or floating point number.
```

```
Yes
```

```
?- atom(ana).
```

```
Yes
```

```
?- atom('Ana').
```

```
Yes
```

```
?- atom(Ana).
```

```
No
```

```
?- atom(5.789).
```

```
No
```

```
?- atomic(5.789).
```

```
Yes
```

```
?- var(X).
```

```
X = _G105
```

```
Yes
```

```
?- X=a,var(X).
```

```
No
```

```
?- nonvar(a).
```

```
Yes
```

Clasificación de términos

- Cuenta ocurrencias

- Definir el predicado `cuenta(A,L,N)` que se verifique si `N` es el número de ocurrencias del átomo `A` en la lista `L`.

- Ejemplos:

```
?- cuenta(a, [a,b,a,a], N).
```

```
N = 3
```

```
Yes
```

```
?- cuenta(a, [a,b,X,Y], N).
```

```
X = _G313
```

```
Y = _G316
```

```
N = 1
```

```
Yes
```

- Programa `cuenta-1.pl`

```
cuenta(_, [], 0).
```

```
cuenta(A, [B|L], N) :-
```

```
    atom(B), A=B, !,
```

```
    cuenta(A, L, M),
```

```
    N is M+1.
```

```
cuenta(A, [B|L], N) :-
```

```
    % not(atom(B), A=B)
```

```
    cuenta(A, L, N).
```

Clasificación de términos

- Si se cambia la definición anterior suprimiendo el literal `atom(B)`...

```
cuenta_1(_, [], 0).
cuenta_1(A, [B|L], N) :-
    A=B, !,
    cuenta_1(A, L, M),
    N is M+1.
cuenta_1(A, [B|L], N) :-
    % not(A=B)
    cuenta_1(A, L, N).
```

- ... entonces, se obtienen los siguientes resultados

```
?- cuenta_1(a, [a,b,a,a], N).
N = 3
Yes
```

```
?- cuenta_1(a, [a,b,X,Y], N).
X = a
Y = a
N = 3 ;
No
```

Construcción de términos

- Los predicados `=..`, `functor`, `args` y `name`

```
Term =.. List
functor(Term,SF,Aridad)
arg(N,Term,Argumento)
name(Term,Lista_ASCII)
```

- Ejemplos

```
?- functor(f(a,b,c),SF,Aridad).
SF = f
Aridad = 3 ;
No
?- Term =.. [padre,juan,luis],functor(Term,SF,Aridad).
Term = padre(juan, luis)
SF = padre
Aridad = 2
Yes
?- Term =.. [padre,juan,luis],arg(N,Term,Argumento).
Term = padre(juan, luis)
N = 1
Argumento = juan ;
Term = padre(juan, luis)
N = 2
Argumento = luis ;
No
```

- Ejemplos:

```
?- name(finidi,L).
L = [102, 105, 110, 105, 100, 105]
Yes
?- name(L,[116, 115, 97, 114, 116, 97, 115]).
L = tsartas
Yes
```


Clasificación de términos

- Figuras proporcionales

- Las figuras geométricas se representan como términos en los que el functor indica el tipo de figura y los argumentos su tamaño. Por ejemplo,

```
cuadrado(3)
triangulo(3,4,5)
circulo(2)
```

- Definir el predicado `alarga(Figura1,Factor,Figura2)` donde `Figura1` y `Figura2` son figuras geométricas del mismo tipo y el tamaño de la `Figura2` es el de la `Figura1` multiplicado por `Factor`.

- Ejemplo

```
?- alarga(triangulo(3,4,5),2,F).
F = triangulo(6, 8, 10)
Yes
```

- Programa `alarga.pl`

```
alarga(Figura1,Factor,Figura2) :-
    Figura1 =.. [Tipo|Argumentos1],
    multiplica_lista(Argumentos1,Factor,Argumentos2),
    Figura2 =.. [Tipo|Argumentos2].

multiplica_lista([],_,[]).
multiplica_lista([X1|L1],F,[X2|L2]) :-
    X2 is X1*F,
    multiplica_lista(L1,F,L2).
```

Clasificación de términos

- **Sustitución de un término**

- Definir el predicado `sustituye(Sub1,Term1,Sub2,Term2)` que se verifique si `Term2` es el término obtenido sustituyendo todas las ocurrencias de `Sub1` en `Term1` por `Sub2`.

- **Ejemplo**

```
?- sustituye(sen(x),2*sen(x)*f(sen(x)),y,T).  
T = 2 * y * f(y)  
Yes
```

```
?- sustituye(a+b,f(a,A+B),c,T).  
A = a  
B = b  
T = f(a, c)  
Yes
```

- **Programa `sustituye.pl`**

```
sustituye(Sub1,Term1,Sub2,Sub2) :-  
    Sub1 = Term1, !.  
sustituye(Sub1,Term1,Sub2,Term1) :-  
    % not(Sub1 = Term1),  
    atomic(Term1), !.  
sustituye(Sub1,Term1,Sub2,Term2) :-  
    % not(Sub1 = Term1),  
    % not(atomic(Term1)),  
    Term1 =.. [Functor|Args1],  
    sustituye_lista(Sub1,Args1,Sub2,Args2),  
    Term2 =.. [Functor|Args2].  
  
sustituye_lista(_, [], _, []).  
sustituye_lista(Sub1, [T1|T1s], Sub2, [T2|T2s]) :-  
    sustituye(Sub1, T1, Sub2, T2),  
    sustituye_lista(Sub1, T1s, Sub2, T2s).
```

Clasificación de términos

- **Términos cerrados**

- Definir el predicado `cerrado(T)` que se verifique `sys T` es un término cerrado (es decir, sin variables).

- **Ejemplo**

```
?- cerrado(f(a+b)).
```

```
Yes
```

```
?- cerrado(f(a+X)).
```

```
No
```

- **Programa `cerrado.pl`**

```
cerrado(T) :-  
    nonvar(T),  
    T =.. [_|Args],  
    cerrados(Args).
```

```
cerrados([]).  
cerrados([X|L]) :-  
    cerrado(X),  
    cerrados(L).
```

- El predicado predefinido `ground` tiene el mismo efecto que `cerrado`.

Clasificación de términos

- El predicado `maplist`

- Definir el predicado

`n_maplist(+Predicado, ?Lista1, ?Lista2)`

que aplique el predicado sobre los sucesivos pares de elementos de la `Lista1` y la `Lista2` y falle si el predicado no puede aplicarse a un par.

- Ejemplos

```
?- n_maplist(succ, [2,4], [3,5]).
```

```
Yes
```

```
?- n_maplist(succ, [2,4], Y).
```

```
Y = [3, 5]
```

```
Yes
```

```
?- n_maplist(succ, X, [3,5]).
```

```
X = [2, 4]
```

```
Yes
```

```
?- n_maplist(succ, [0,4], [3,5]).
```

```
No
```

- Programa `n_maplist.pl`

```
n_maplist(_, [], []).
```

```
n_maplist(F, [X1|L1], [X2|L2]) :-
```

```
    G =.. [F,X1,X2],
```

```
    G,
```

```
    n_maplist(F,L1,L2).
```

- El predicado predefinido `maplist` tiene el mismo efecto que `n_maplist`.

Clasificación de términos

- **Números de Turing**

- Un número entero es de Turing si es igual al cuadrado de la suma de sus cifras. Definir la relación `numero_de_turing(N)` que se verifique si `N` es un número de Turing y calcular los números de Turing menores que 1.000.

- Programa `turing.pl`

```
numero_de_turing(N) :-  
    cifras(N,L),  
    suma_cifras(L,M),  
    N ::= M**2.
```

```
cifras(N,L) :-  
    name(N,L1),  
    reduce(L1,L).
```

```
reduce([X|R],[Y|S]) :-  
    Y is X - 48,  
    reduce(R,S).  
reduce([],[]).
```

```
suma_cifras([X|R],N) :-  
    suma_cifras(R,N1),  
    N is X+N1.  
suma_cifras([],0).
```

- **Sesión**

```
?- between(0,1000,N),numero_de_turing(N).  
N = 0 ;  
N = 1 ;  
N = 81 ;  
No
```

Varias clases de igualdad

- Unificación =, \=

```
?- f(a,X)=f(Y,b).
```

```
X = b
```

```
Y = a ;
```

```
No
```

```
?- [X,Y,Z]=[Y,Z,X].
```

```
X = _G159
```

```
Y = _G159
```

```
Z = _G159 ;
```

```
Nos
```

```
?- f(a,b)\=f(_X,_X).
```

```
Yes
```

- Test de ocurrencia

- Programa monstruo.pl

```
monstruo :- X=f(X).
```

- Sesión

```
?- monstruo.
```

```
Yes
```

Varias clases de igualdad

- Aritmética `is`, `:=`, `=` `\=`

```
?- X is 3+2.
```

```
X = 5 ;
```

```
No
```

```
?- 5 is 3+X.
```

```
[WARNING: Arguments are not sufficiently instantiated]
```

```
^ Exception: ( 7) 5 is 3+_G118 ? creep
```

```
[WARNING: Unhandled exception]
```

```
?- 11 - 3 := 2^3.
```

```
Yes
```

```
?- 11 - 3 := 2^X.
```

```
[WARNING: Arguments are not sufficiently instantiated]
```

```
Exception: ( 7) 11-3:=2^_G157 ? creep
```

```
[WARNING: Unhandled exception]
```

```
?- 11 - 3 =\= 2^7.
```

```
Yes
```

```
?- 11 - X =\= 2^7.
```

```
[WARNING: Arguments are not sufficiently instantiated]
```

```
Exception: ( 7) 11-_G154=\=2^7 ? creep
```

```
[WARNING: Unhandled exception]
```

Varias clases de igualdad

- Identidad `==`, `\==`

?- `f(a,b)==f(a,b)` .

Yes

?- `f(a,b)==f(a,X)` .

No

?- `f(a,Y)==f(a,X)` .

No

?- `X \== Y` .

`X = _G111`

`Y = _G112`

Yes

?- `_X \== _Y` .

Yes

?- `f(A,g(B,i),Z)==f(A,g(B,i),Z)` .

`A = _G240`

`B = _G237`

`Z = _G242`

Yes

Varias clases de igualdad

- Cuenta ocurrencias (II)

- Definir el predicado `cuenta(A,L,N)` que se verifique si `N` es el número de ocurrencias idénticas al átomo `A` en la lista `L`.

- Ejemplos

```
?- cuenta(a, [a,b,a,a], N).
```

```
N = 3
```

```
Yes
```

```
?- cuenta(a, [a,b,X,Y], N).
```

```
X = _G313
```

```
Y = _G316
```

```
N = 1
```

```
Yes
```

- Programa `cuenta-2.pl`

```
cuenta(_, [], 0).
```

```
cuenta(A, [B|L], N) :-
```

```
    A == B,
```

```
    cuenta(A, L, M),
```

```
    N is M+1.
```

```
cuenta(A, [B|L], N) :-
```

```
    A \== B,
```

```
    cuenta(A, L, N).
```

Bibliografía

- Bratko, I. *Prolog Programming for Artificial Intelligence (2nd ed.)* (Addison–Wesley, 1990)
 - Cap. 2: “Syntax and Meaning of Prolog Programs”
 - Cap. 6: “Input and Output”
 - Cap. 7: “More Built–in Procedures”
- Clocksin, W.F. y Mellish, C.S. *Programming in Prolog (Fourth Edition)* (Springer Verlag, 1994)
 - Cap. 2: “A closer look”